Program 13 Date 10 Nov 2012

Aim:

Write assembly language macros:

1. To read a character from the keyboard
2. To display a character
3. Use the above two functions to read a string of characters from the keyboard terminated by the carriage return and prints the string on the display in the next line.

Theory:

The program uses the 01h and 02h service routines of the 21h interrupt to accept or write a single character. A macro is created using these instructions used in series to accept a string.

Source code:

.8086

.MODEL SMALL

.STACK 512

.DATA

CRLF DB 13,10,'$'

STRING DB 64 DUP(?)

.CODE

START:

MOV AX,@DATA

MOV DS,AX

READCHAR MACRO

MOV AX,0100H

INT 21H

ENDM

WRITECHAR MACRO

;CHAR TO BE MOVED TO DL

MOV AX,0200H

INT 21H

ENDM

READSTR MACRO

MOV SI,0

RST:

READCHAR

MOV STRING[SI],AL

INC SI

CMP AL,13

JNZ RST

MOV STRING[SI],'$'

ENDM

WRITESTR MACRO

MOV SI,0

WST:

MOV DL,STRING[SI]

WRITECHAR

INC SI

CMP DL,13

JNZ WST

ENDM

READCHAR

MOV DL,AL

WRITECHAR

LEA DX,CRLF

MOV AH,09H

INT 21H

READSTR

LEA DX,CRLF

MOV AH,09H

INT 21H

WRITESTR

MOV AH,4CH

INT 21H

END START

Conclusion:

An 8086 assembly language macro to read/write a character and strings was written, assembled, linked and debugged to obtain expected output.

­­Program 14 Date 10 Nov 2012

Aim:

Write assembly language to read an alphanumeric character and display its equivalent ASCII code at the centre of the screen.

Theory:

The program uses interrupt service routines to carry out tasks like moving the cursor position, clearing the screen etc. The 8086 assembly language provides an extensive list of interrupt services to perform a wide list of functions. The two interrupts used here are 10h and 21h. The former controls the output device, in this case, the monitor. The latter controls the input/output streams and buffers for the monitor and the keyboard.

Source code:

.8086

.MODEL SMALL

.STACK 512

.DATA

CHAR DB ?

INTEG DB 4 DUP(0)

.CODE

START:

MOV AX,@DATA

MOV DS,AX

MOV AH,01H

INT 21H

MOV CHAR,AL

;INVOKE VIDEO INTERRUPT

;CLEAR SCREEN

MOV AL,00H

MOV CX,00H

MOV DX,1850H

MOV AH,06H

MOV BH,07H

INT 10H

;SET CURSOR POSITION

MOV DX,0C23H

MOV AH,02H

MOV BH,00H

INT 10H

MOV [INTEG+3],'$'

MOV AL,CHAR

MOV CL,0AH

MOV BX,02H

LOOP0:

MOV AH,00H

DIV CL

ADD AH,'0'

MOV INTEG[BX],AH

DEC BX

JNS LOOP0

LEA DX,INTEG

MOV AH,09H

INT 21H

MOV AH,08H

INT 21H

MOV AH,4CH

INT 21H

END START

Conclusion:

An 8086 assembly language macro to read a character and display its equivalent ASCII value in the centre of the screen was written, assembled, linked and debugged to obtain expected output.

­­ Program 12 Date 26 September 2012

Aim:

To write 8086/8087 program to compute the real roots of the quadratic equation.

Theory:

The quadratic equation being solved is 2x2 + 5x + 3 = 0. This is done by using the quadratic formula

where the values of *x* are the roots of the given equation. This gives two possible values of *x*. These values are stored in variables x and y respectively. This discriminant is stored in the variable disc.

Source code:

.8087

.MODEL SMALL

.STACK 512

.DATA

A DW 2.0

B DW 5.0

C DW 3.0

DISC DW ?

X DW ?

Y DW ?

.CODE

START:

MOV AX,@DATA

MOV DS,AX

FINIT

FLD B

FLD B

FMUL

FLD A

FLD C

FMUL

FLD 4.0

FMUL

FSUB

FST DISC

FINIT

FLD B

FLD -1.0

FMUL

FLD DISC

FSUB

FLD A

FLD 2.0

FMUL

FDIV

FST X

FINIT

FLD B

FLD -1.0

FMUL

FLD DISC

FADD

FLD A

FLD 2.0

FMUL

FDIV

FST Y

MOV AH,4CH

INT 21H

END START

Conclusion:

An 8086/8087 assembly language program to calculate the roots of a quadratic equation was written, assembled, linked and debugged to obtain expected output.

Program 15 Date 10 Nov 2012

Aim:

Write assembly language to reverse a string and check whether it is a palindrome or not.

Theory:

A palindrome is a word, phrase, number, or other sequence of units that may be read the same way in either direction. The program checks for the inbuilt strng for being a palindrome and displays the result.

Source code:

.8086

.MODEL SMALL

.STACK 512

.DATA

STRNG DB "RACECAR",'$'

STRNGLEN DW $-STRNG

STRNGREV DB 20 DUP(' '),13,10

PROMPT0 DB 13,10,"PALINDROME",'$'

PROMPT1 DB 13,10,"NOT A PALINDROME",'$'

.CODE

START:

MOV AX,@DATA

MOV DS,AX

MOV ES,AX

MOV CX,STRNGLEN

ADD CX,-2

LEA SI,STRNG

LEA DI,STRNGREV

ADD SI,STRNGLEN

ADD SI,-2

LOOP0:

MOV AL,[SI]

MOV [DI],AL

DEC SI

INC DI

LOOP LOOP0

MOV AL,[SI]

MOV [DI],AL

INC DI

MOV DL,'$'

MOV [DI], DL

MOV CX,STRNGLEN

LEA DX,PROMPT1

MOV AH,09H

INT 21H

LEA DX,STRNGREV

MOV AH,09H

INT 21H

PCHECK:

LEA SI,STRNG

LEA DI,STRNGREV

REPE CMPSB

JNE PFALSE

PTRUE:

MOV AH,09H

LEA DX,PROMPT0

INT 21H

JMP TERMINATE

PFALSE:

MOV AH,09H

LEA DX,PROMPT2

INT 21H

TERMINATE:

MOV AX,4C00H

INT 21H

END START

Conclusion:

An 8086 assembly language program to determine if a string is a palindrome or not was written, assembled, linked and debugged to obtain expected output.

Program 16 Date 10 Nov 2012

Aim:

To write an assembly language program to read two strings, store them in locations STR1 and STR2 and check whether they are equal or not and display appropriated messages. Also display the length of the stored strings.

Source code:

. 8086

.MODEL SMALL

.STACK 512

.DATA

STRNG1 DB 64 DUP('$')

STRNG2 DB 64 DUP('$')

STRNG1\_LEN DW $-STRNG1

STRNG2\_LEN DW $-STRNG2

CRLF DB 13,10,'$'

PROMPT1 DB 10,13,"STRING 1: ",'$'

PROMPT2 DB 10,13,"STRING 2: ",'$'

PROMPT3 DB 10,13,"STRINGS ARE EQUAL",'$'

PROMPT4 DB 10,13,"STRINGS ARE NOT EQUAL",'$'

.CODE

EXTRN READSINT:NEAR,WRITESINT:NEAR

START:

MOV AX,@DATA

MOV DS,AX

READSTRNG MACRO STRNG

MOV AH,0AH

LEA DX,STRNG

INT 21H

ENDM

WRITESTRNG MACRO STRNG

MOV AH,09H

LEA DX,STRNG

INT 21H

ENDM

WRITESTRNG PROMPT1

READSTRNG STRNG1

WRITESTRNG PROMPT2

READSTRNG STRNG2

LEA DX,CRLF

MOV AH,09H

INT 21H

MOV AX,STRNG1\_LEN

CALL WRITESINT

LEA DX,CRLF

MOV AH,09H

INT 21H

MOV AX,STRNG2\_LEN

CALL WRITESINT

LEA SI,STRNG1

LEA DI,STRNG2

MOV CL,STRNG1+1

MOV CH,00H

REPE CMPSB

JNE NOTEQUAL

WRITESTRNG PROMPT3

JMP TERMINATE

NOTEQUAL:

WRITESTRNG PROMPT4

TERMINATE:

MOV AX,4C00H

INT 21H

END START

Conclusion:

An 8086 assembly language program to determine if two read strings were equal or not was written, assembled, linked and debugged to obtain expected output.

Program 17 Date 10 Nov 2012

Aim:

To write an assembly language program to read a name from the keyboard and display it at a specified location on the screen in front of the message ‘WHAT IS YOUR NAME?’. The entire screen has to be cleared before display.

Source code:

.8086

.MODEL SMALL

.STACK 512

.DATA

PROMPT1 DB "WHAT IS YOUR NAME?",'$'

PROMPT2 DB "YOU ARE ",’$’

NAME0 DB 99 DUP(0)

.CODE

START:

MOV AX,@DATA

MOV DS,AX

POS PROC

MOV AH,02H

MOV BH,00H

INT 10H

RET

POS ENDP

MOV AH,06H

MOV AL,00H

MOV BH,07H

MOV CX,00H

MOV DX,1850H

INT 10H

MOV DX,0C23H

CALL POS

LEA DX,PROMPT1

MOV AH,09H

INT 21H

LEA SI,NAME0

LEA DI,NAME0

LOOP1:

MOV AH,01H

INT 21H

CMP AL,08H

JE LOOP2

CMP AL,0DH

JE TAG1

MOV [SI],AL

INC SI

JMP LOOP1

LOOP2:

MOV DL,' '

MOV AH,02H

INT 21H

MOV DL,08H

INT 21H

CMP SI,DI

JE LOOP1

DEC SI

JMP LOOP1

TAG1:

MOV AL,'$'

MOV [SI],AL

MOV DX,0D23H

CALL POS

LEA DX,PROMPT2

MOV AH,09H

INT 21H

MOV AH,4CH

INT 21H

END START

Conclusion:

An 8086 assembly language program to read a name and write it according to the given requirements was written, assembled, linked and debugged to obtain expected output.

Program 18 Date 10 Nov 2012

Aim:

To write an assembly language program to compute nCr using a recursive procedure. Assume that 'n' and 'r' are non-negative integers

Theory:

In mathematics a combination is a way of selecting several things out of a larger group, where (unlike permutations) order does not matter. It is defined as

The program defines the factorial as a procedure and calculates it recursively.

Source code:

.8086

.MODEL SMALL

.DATA

N DB 05H

R DB 02H

NCR DW ?

.CODE

EXTRN READSINT:NEAR,WRITESINT:NEAR

START:

MOV AX,@DATA

MOV DS,AX

NCRP PROC

CMP AX,BX

JE TAG1

CMP BX,00H

JE TAG1

CMP BX,01H

JE TAG2

DEC AX

CMP AX,BX

JE TAG3

PUSH AX

PUSH BX

CALL NCRP

POP BX

POP AX

DEC BX

PUSH AX

PUSH BX

CALL NCRP

POP BX

POP AX

RET

TAG1:

INC NCR

RET

TAG2:

ADD NCR,AX

RET

TAG3:

ADD NCR,AX

INC NCR

RET

NCRP ENDP

MOV AX,00H

MOV AL,N

MOV BL,R

MOV NCR,00H

CALL NCRP

MOV AH,4CH

INT 21H

END START

Conclusion:

An 8086 assembly language program to compute nCr using a recursive procedure was written, assembled, linked and debugged to obtain expected output.

Program 19 Date 10 Nov 2012

Aim:

To write an assembly language program to read the current time from the system and display it in the standard format on the screen.

Theory:

The program uses 2ch service of 21h interrupt to obtain the system time. The interrupt stores hours in CH, minutes in CL, seconds in DH and 1/100 seconds in DL. The values are converted into character strings before displayed on the screen.

Source code:

. 8086

.MODEL SMALL

.STACK 512

.DATA

.CODE

START:

MOV AX,@DATA

MOV DS,AX

MOV AH,2CH

INT 21H

MOV BL,0AH

MOV AL,CH

CALL ATIME

MOV AL,CL

CALL ATIME

MOV AL,DH

CALL MTIME

MOV AH,4CH

INT 21H

ATIME PROC

CALL MTIME

MOV DL,':'

MOV AH,02H

INT 21H

RET

ATIME ENDP

MTIME PROC

MOV AH,00H

DIV BL

MOV DL,'0'

XCHG AL,AH

ADD DL,AH

MOV AH,02H

PUSH AX

INT 21H

POP AX

MOV DL,AL

ADD DL,'0'

INT 21H

RET

MTIME ENDP

END START

Conclusion:

An 8086 assembly language program to read the current time from the system and display it in the standard format on the screen was written, assembled, linked and debugged to obtain expected output.

Program 20 Date 10 Nov 2012

Aim:

To write an assembly language program to simulate a decimal up-counter to display 00-99.

Source code:

. .8086

.MODEL SMALL

.STACK 512

.DATA

COUNTER DB 64H

.CODE

START:

MOV AX,@DATA

MOV DS,AX

MOV CL,COUNTER

MOV AL,00H

LOOP0:

MOV AL,64H

SUB AL,CL

MOV BL,0AH

MOV AH,00H

DIV BL

XCHG AL,AH

MOV DL,AH

ADD DL,'0'

MOV AH,02H

PUSH AX

INT 21H

POP AX

MOV DL,AL

ADD DL,'0'

INT 21H

MOV DL,0DH

INT 21H

PUSH CX

MOV BX,01AAH

LOOP1:

LOOP LOOP1

DEC BX

JNZ LOOP1

POP CX

LOOP LOOP0

MOV AH,4CH

INT 21H

END START

Conclusion:

An 8086 assembly language program to simulate a decimal up-counter was written, assembled, linked and debugged to obtain expected output.